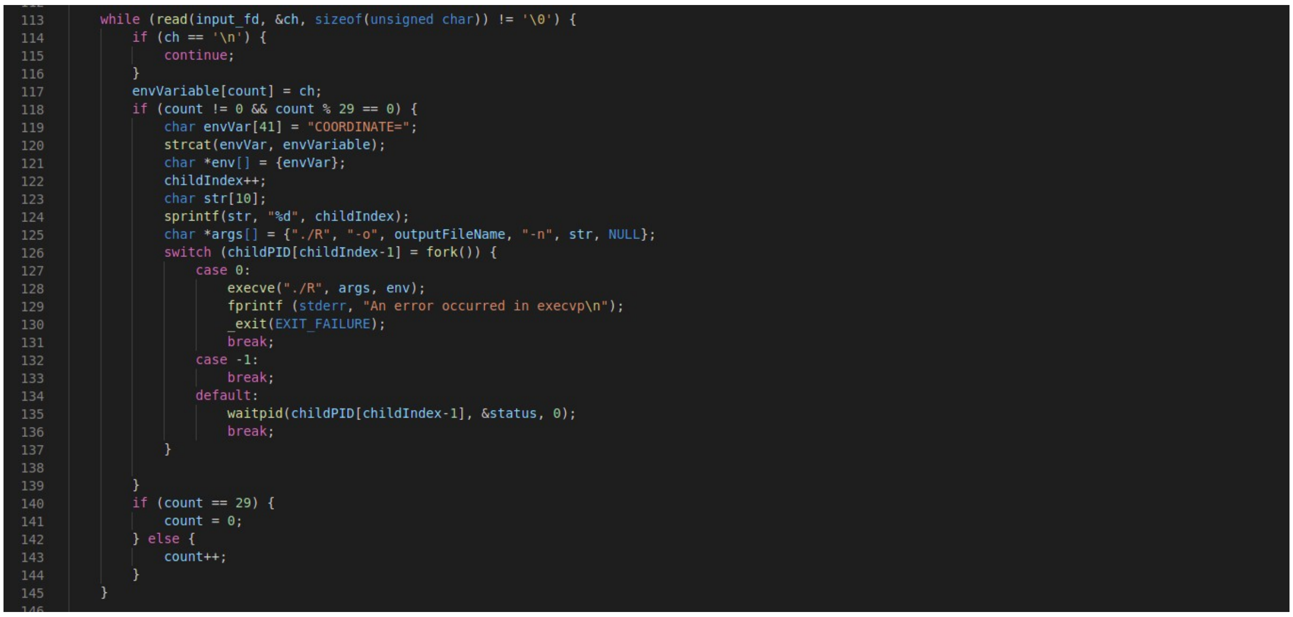
Burak Tekdamar

161044115

CSE 344 HW2 REPORT

First of all, I read the input file once and kept a counter named NUM_PROCESS to find out how many processes can occur. Then I re-read the file and call a fork() function for every 30 characters in the file. I wrote every character I got from the file into a string named “env” so that I could send it to the child processes. Then I called the execve() function and sent the output file name, the order of creation and the environment variable named “env” as function parameters to the program named R. I exported the environment variable I sent to the child process and found the covariance matrix with the incoming coordinates and wrote the results to the output file. I locked it with the fcntl() function so that no other process can write to the file during the write process. The data obtained from each operation takes up one line in the output file.

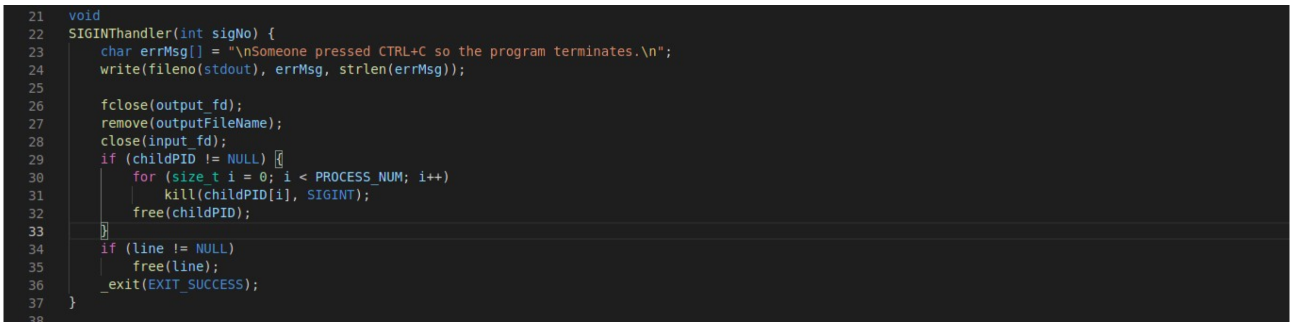
Fork() + execve()



```c
    while (read(input_fd, &ch, sizeof(unsigned char)) != '\0') {
        if (ch == '\n') {
            continue;
        }
        envVariable[count] = ch;
        if (count != 0 && count % 29 == 0) {
            char envVar[41] = "COORDINATE=";
            strcat(envVar, envVariable);
            char *env[] = {envVar};
            childIndex++;
            char str[10];
            sprintf(str, "%d", childIndex);
            char *args[] = {"./R", "-o", outputFileName, "-n", str, NULL};
            switch (childPID[childIndex-1] = fork()) {
                case 0:
                    execve("./R", args, env);
                    fprintf (stderr, "An error occurred in execvp\n");
                    _exit(EXIT_FAILURE);
                    break;
                case -1:
                    break;
                default:
                    waitpid(childPID[childIndex-1], &status, 0);
                    break;
            }

        }
        if (count == 29) {
            count = 0;
        } else {
            count++;
        }
    }
```

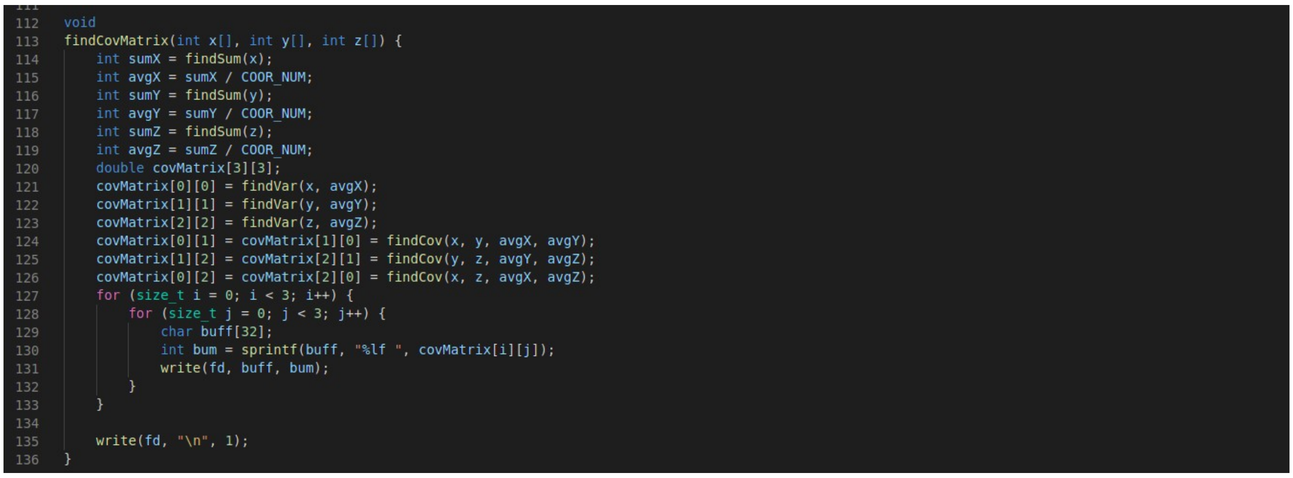
I made the parent process wait until a created child process terminates. When it comes to the end of the input file, I read the output file line by line in the parent process to calculate the Frobenius norm of the matrices written to the output file by the child processes. I calculated the norms of the matrices in each row with the function called findFrobenius(). Then I wrote a function called findClosest() to find the two matrices closest to each other from the obtained norm values. I wrote a handler to capture the SIGINT signal from the parent process and free all resources, close open files, delete output file, terminate all open child processes.

SIGINT handler



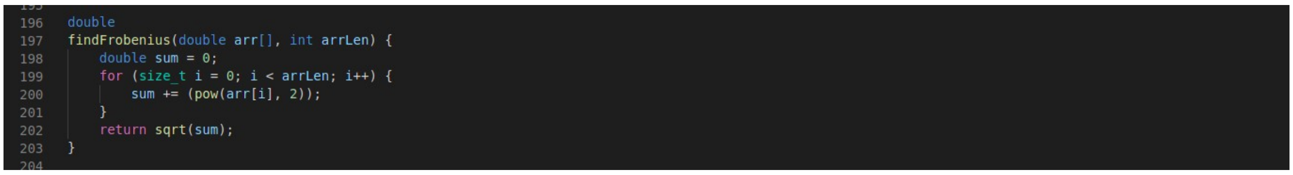
```c
void
SIGINThandler(int sigNo) {
    char errMsg[] = "\nSomeone pressed CTRL+C so the program terminates.\n";
    write(fileno(stdout), errMsg, strlen(errMsg));

    fclose(output_fd);
    remove(outputFileName);
    close(input_fd);
    if (childPID != NULL) {
        for (size_t i = 0; i < PROCESS_NUM; i++)
            kill(childPID[i], SIGINT);
        free(childPID);
    }
    if (line != NULL)
        free(line);
    _exit(EXIT_SUCCESS);
}
```

Covariance Matrix Finder



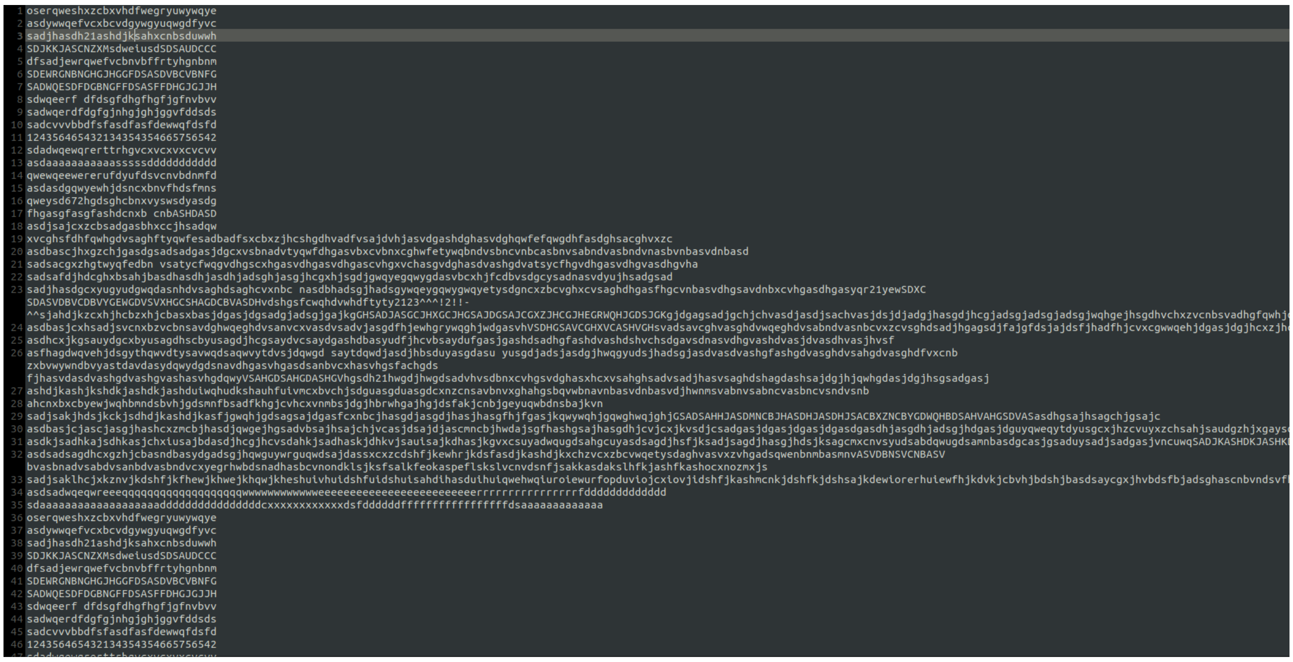
```c
void
findCovMatrix(int x[], int y[], int z[]) {
    int sumX = findSum(x);
    int avgX = sumX / COOR_NUM;
    int sumY = findSum(y);
    int avgY = sumY / COOR_NUM;
    int sumZ = findSum(z);
    int avgZ = sumZ / COOR_NUM;
    double covMatrix[3][3];
    covMatrix[0][0] = findVar(x, avgX);
    covMatrix[1][1] = findVar(y, avgY);
    covMatrix[2][2] = findVar(z, avgZ);
    covMatrix[0][1] = covMatrix[1][0] = findCov(x, y, avgX, avgY);
    covMatrix[1][2] = covMatrix[2][1] = findCov(y, z, avgY, avgZ);
    covMatrix[0][2] = covMatrix[2][0] = findCov(x, z, avgX, avgZ);
    for (size_t i = 0; i < 3; i++) {
        for (size_t j = 0; j < 3; j++) {
            char buff[32];
            int bum = sprintf(buff, "%lf ", covMatrix[i][j]);
            write(fd, buff, bum);
        }
    }

    write(fd, "\n", 1);
}
```

Frobenius Norm Finder



```c
double
findFrobenius(double arr[], int arrLen) {
    double sum = 0;
    for (size_t i = 0; i < arrLen; i++) {
        sum += (pow(arr[i], 2));
    }
    return sqrt(sum);
}
```

Input File

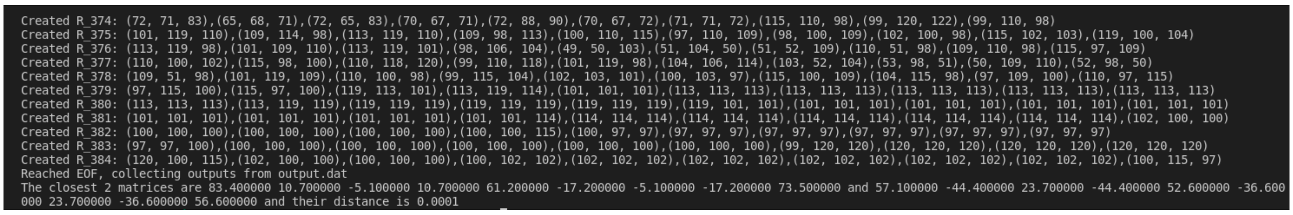


```
oserqweshxzcbxvhdfwegryuwywqye
asdywwqefvcxbcvdgywgyuqwgdfyvc
sadjhasdh21ashdjksahxcnbsduwwh
SDJKKJASCNZXMsdweiusdSDSAUDCCC
dfsadjewrqwefvcbnvbffrtyhgnbnm
SDEWRGNBNGHGJHGGFDSASDVBCVBNFG
SADWQESDFDGBNGFFDSASFFDHGJGJJH
sdwqeerf dfdsgfdhgfhgfjgfnvbvv
sadwqerdfdgfgjnhgjghjggvfddsds
sadcvvvbbdfsfasdfasfdewwqfdsfd
124356465432134354354665756542
sdadwqewqrerttrhgvcxvcxvxcvcvv
asdaaaaaaaaaaassssssdddddddddd
qwewqeewererufdyufdsvcnvbdnmfd
asdasdgqwyewhjdsncxbnvfhdsfmns
qweysd672hgdsghcbnxvyswsdyasdg
fhgasgfasgfashdcnxb cnbASHDASD
asdjsajcxzcbsadgasbhxccjhsadqw
xvcghsfdhfqwhgdvsaghftyqwfesadbadfsxcbxzjhcshgdhvadfvsajdvhjasvdgashdghasvdghqwfefqwgdhfasdghsacghvxzc
asdbascjhxgzchjgasdgsadsadgasjdgcxvsbnadvtyqwfdhgasvbxcvbnxcghwfetywqbndvsbncvnbcasbnvsabndvasbndvnasbvnbasvdnbasd
sadsacgxzhgtwyqfedbn vsatycfwqgvdhgscxhgasvdhgasvdhgascvhgxvchasgvdghasdvashgdvatsycfhgvdhgasvdhgvasdhgvha
sadsafdjhdcghxbsahjbasdhasdhjasdhjadsghjasgjhcgxhjsgdjgwqyegqwygdasvbcxhjfcdbvsdgcysadnasvdyujhsadgsad
sadjhasdgcxyugyudgwqdasnhdvsaghdsaghcvxnbc nasdbhadsgjhadsgywqeygqwygwqyetysdgncxzbcvghxcvsaghdhgasfhgcvnbasvdhgsavdnbxcvhgasdhgasyqr21yewSDXC
SDASVDBVCDBVYGEWGDVSVXHGCSHAGDCBVASDHvdshgsfcwqhdvwhdftyty2123^^^!2!!-
^^sjahdjkzcxhjhcbzxhjcbasxbasjdgasjdgsadgjadsgjgajkgGHSADJASGCJHXGCJHGSAJDGSAJCGXZJHCGJHEGRWQHJGDSJGKgjdgagsadjgchjchvasdjasdjsachvasjdsjdjadgjhasgdjhcgjadsgjadsgjadsgjwqhgejhsgdhvchxzvcnbsvadhgfqwhj
asdbasjcxhsadjsvcnxbzvcbnsavdghwqeghdvsanvcxvasdvsadvjasgdfhjewhgrywqghjwdgasvhVSDHGSAVCGHXVCASHVGHsvadsavcghvasghdvwqeghdvsabndvasnbcvxzcvsghdsadjhgagsdjfajgfdsjajdsfjhadfhjcvxcgwwqehjdgasjdgjhcxzjh
asdhcxjkgsauydgcxbyusagdhscbyusagdjhcgsaydvcsaydgashdbasyudfjhcvbsaydufgasjgashdsadhgfashdvashdshvchsdgavsdnasvdhgvashdvasjdvasdhvasjhvsf
asfhagdwqvehjdsgythqwvdtysavwqdsaqwvytdvsjdqwgd saytdqwdjasdjhbsduyasgdasu yusgdjadsjasdgjhwqgyudsjhadsgjasdvasdvashgfashgdvasghdvsahgdvasghdfvxcnb
zxbvwywndbvyastdavdasydqwydgdsnavdhgasvhgasdsanbvcxhasvhgsfachgds
fjhasvdasdvashgdvashgvashasvhgdqwyVSAHGDSAHGDASHGVhgsdh21hwgdjhwgdsadvhvsdbnxcvhgsvdghasxhcxvsahghsadvsadjhasvsaghdshagdashsajdgjhjqwhgdasjdgjhsgsadgasj
ashdjkashjkshdkjashdkjashduiwqhudkshauhfuivmcxbvchjsdguasgduasgdcxnzcnsavbnvxghahgsbqvwbnavnbasvdnbasvdjhwnmsvabnvsabncvasbncvsndvsnb
ahcnxbxcbyewjwqhbmndsbvhjgdsmnfbsadfkhgjcvhcxvnmbsjdgjhbrwhgajhgjdsfakjcnbjgeyuqwbdnsbajkvn
sadjsakjhdsjkckjsdhdjkashdjkasfjgwqhjgdsagsajdgasfcxnbcjhasgdjasgdjhasjhasgfhjfgasjkqwywqhjgqwghwqjghjGSADSAHHJASDMNCBJHASDHJASDHJSACBXZNCBYGDWQHBDSAHVAHGSDVASasdhgsajhsagchjgsajc
asdbasjcjascjasgjhashcxzmcbjhasdjqwgejhgsadvbsajhsajchjvcasjdsajdjascmncbjhwdajsgfhashgsajhasgdhjcvjcxjkvsdjcsadgasjdgasjdgasjdgasdgasdhjasgdhjadsgjhdgasjdguyqweqytdyusgcxjhzcvuyxzchsahjsaudgzhjxgays
asdkjsadhkajsdhkasjchxiusajbdasdjhcgjhcvsdahkjsadhaskjdhkvjsauisajkdhasjkgvxcsuyadwqugdsahgcuyasdsagdjhsfjksadjsagdjhasgjhdsjksagcmxcnvsyudsabdqwugdsamnbasdgcasjgsaduysadjsadgasjvncuwqSADJKASHDKJASHK
asdsadsagdhcxgzhjcbasndbasydgadsgjhqwguywrguqwdsajdassxcxzcdshfjkewhrjkdsfasdjkashdjkxchzvcxzbcvwqetysdaghvasvxzvhgadsqwenbnmbasmnvASVDBNSVCNBASV
bvasbnadvsabdvsanbdvasbndvcxyegrhwbdsnadhasbcvnondklsjksfsalkfeokaspeflskslvcnvdsnfjsakkasdakslhfkjashfkashocxnozmxjs
sadjsaklhcjxkznvjkdshfjkfhewjkhwejkhqwjkheshuivhuidshfuidshuisahdihasduihuiqwehwqiuroiewurfopduviojcxiovjidshfjkashmcnkjdshfkjdshsajkdewiorerhuiewfhjkdvkjcbvhjbdshjbasdsaycgxjhvbdsfbjadsghascnbvndsvf
asdsadwqeqwreeeqqqqqqqqqqqqqqqqqqwwwwwwwwwwwweeeeeeeeeeeeeeeeeeeeeeeeerrrrrrrrrrrrrrfdddddddddddd
sdaaaaaaaaaaaaaaaaaaadddddddddddddddddcxxxxxxxxxxxxdsfddddddffffffffffffffffdsaaaaaaaaaaaaa
oserqweshxzcbxvhdfwegryuwywqye
asdywwqefvcxbcvdgywgyuqwgdfyvc
sadjhasdh21ashdjksahxcnbsduwwh
SDJKKJASCNZXMsdweiusdSDSAUDCCC
dfsadjewrqwefvcbnvbffrtyhgnbnm
SDEWRGNBNGHGJHGGFDSASDVBCVBNFG
SADWQESDFDGBNGFFDSASFFDHGJGJJH
sdwqeerf dfdsgfdhgfhgfjgfnvbvv
sadwqerdfdgfgjnhgjghjggvfddsds
sadcvvvbbdfsfasdfasfdewwqfdsfd
124356465432134354354665756542
```

OUTPUTS

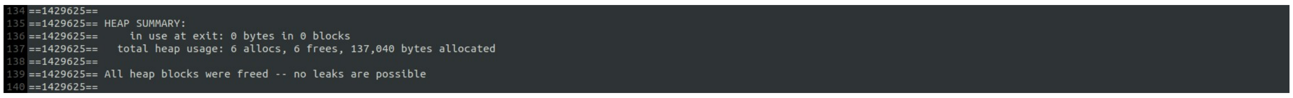
Usage: ./P -i inputFileName -o outputFileName

Terminal screen



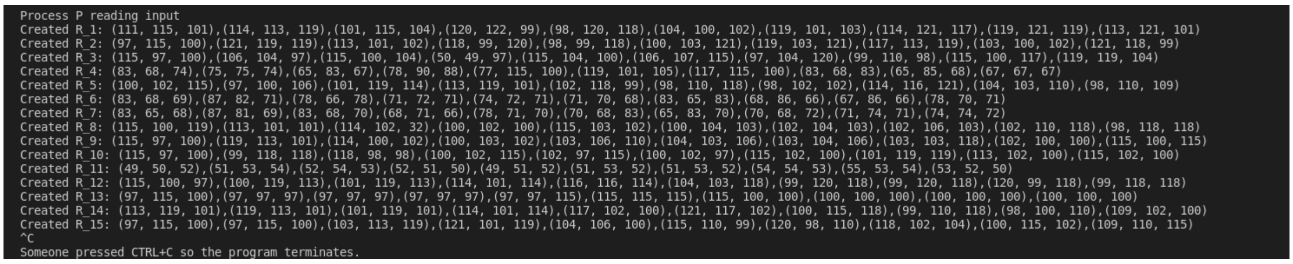
```
Created R_374: (72, 71, 83),(65, 68, 71),(72, 65, 83),(70, 67, 71),(72, 88, 90),(70, 67, 72),(71, 71, 72),(115, 110, 98),(99, 120, 122),(99, 110, 98)
Created R_375: (101, 119, 110),(109, 114, 98),(113, 119, 110),(109, 98, 113),(100, 110, 115),(97, 110, 109),(98, 100, 109),(102, 100, 98),(115, 102, 103),(119, 100, 104)
Created R_376: (113, 119, 98),(101, 109, 110),(113, 119, 101),(98, 106, 104),(49, 50, 103),(51, 104, 50),(51, 52, 109),(110, 51, 98),(109, 110, 98),(115, 97, 109)
Created R_377: (110, 100, 102),(115, 98, 100),(110, 118, 120),(99, 110, 118),(101, 119, 98),(104, 106, 114),(103, 52, 104),(53, 98, 51),(50, 109, 110),(52, 98, 50)
Created R_378: (109, 51, 98),(101, 119, 109),(110, 100, 98),(99, 115, 104),(102, 103, 101),(100, 103, 97),(115, 100, 109),(104, 115, 98),(97, 109, 100),(110, 97, 115)
Created R_379: (97, 115, 100),(115, 97, 100),(119, 113, 101),(113, 119, 114),(101, 101, 101),(113, 113, 113),(113, 113, 113),(113, 113, 113),(113, 113, 113),(113, 113, 113)
Created R_380: (113, 113, 113),(113, 119, 119),(119, 119, 119),(119, 119, 119),(119, 119, 119),(119, 101, 101),(101, 101, 101),(101, 101, 101),(101, 101, 101),(101, 101, 101)
Created R_381: (101, 101, 101),(101, 101, 101),(101, 101, 101),(101, 101, 114),(114, 114, 114),(114, 114, 114),(114, 114, 114),(114, 114, 114),(114, 114, 114),(102, 100, 100)
Created R_382: (100, 100, 100),(100, 100, 100),(100, 100, 100),(100, 100, 115),(100, 97, 97),(97, 97, 97),(97, 97, 97),(97, 97, 97),(97, 97, 97),(97, 97, 97)
Created R_383: (97, 97, 100),(100, 100, 100),(100, 100, 100),(100, 100, 100),(100, 100, 100),(100, 100, 100),(99, 120, 120),(120, 120, 120),(120, 120, 120),(120, 120, 120)
Created R_384: (120, 100, 115),(102, 100, 100),(100, 100, 100),(100, 102, 102),(102, 102, 102),(102, 102, 102),(102, 102, 102),(102, 102, 102),(102, 102, 102),(100, 115, 97)
Reached EOF, collecting outputs from output.dat
The closest 2 matrices are 83.400000 10.700000 -5.100000 10.700000 61.200000 -17.200000 -5.100000 -17.200000 73.500000 and 57.100000 -44.400000 23.700000 -44.400000 52.600000 -36.600
000 23.700000 -36.600000 56.600000 and their distance is 0.0001
```

valgrind --leak-check=full --show-leak-kinds=all --track-origins=yes --verbose --log-file=valgrind-out.txt ./P -i input -o output.dat



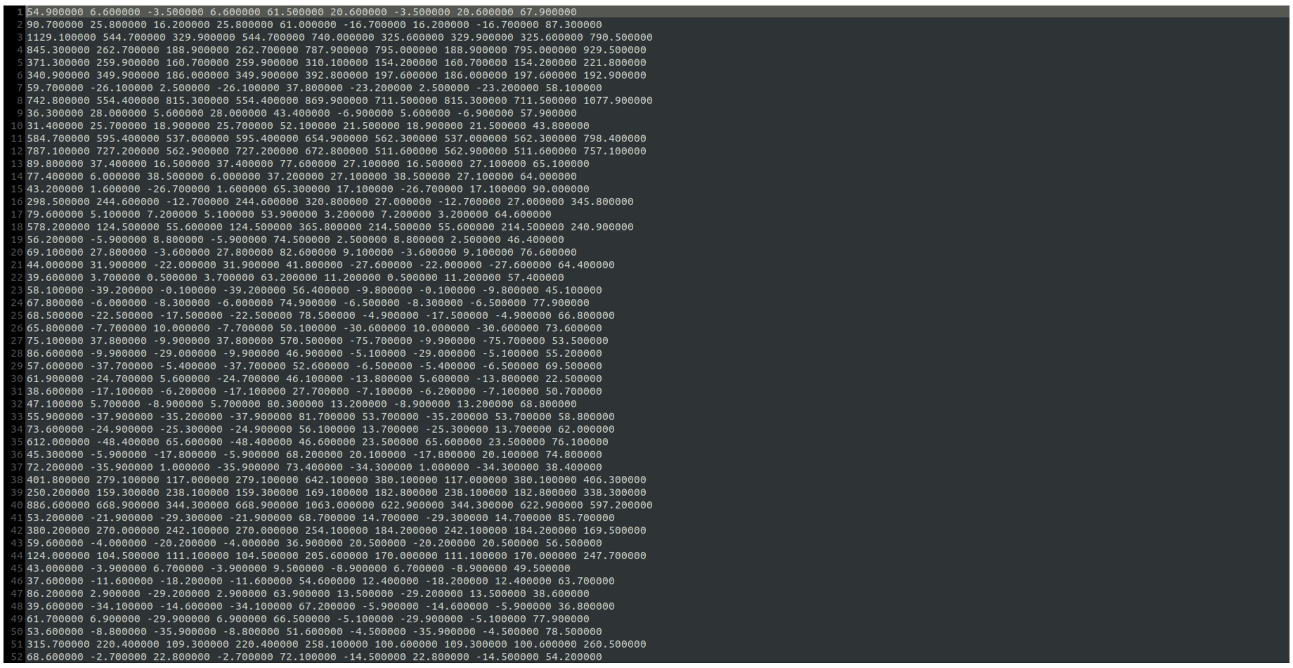
```
==1429625==
==1429625== HEAP SUMMARY:
==1429625==     in use at exit: 0 bytes in 0 blocks
==1429625==   total heap usage: 6 allocs, 6 frees, 137,040 bytes allocated
==1429625==
==1429625== All heap blocks were freed -- no leaks are possible
==1429625==
```

SIGINT Result



```
Process P reading input
Created R_1: (111, 115, 101),(114, 113, 119),(101, 115, 104),(120, 122, 99),(98, 120, 118),(104, 100, 102),(119, 101, 103),(114, 121, 117),(119, 121, 119),(113, 121, 101)
Created R_2: (97, 115, 100),(121, 119, 119),(113, 101, 102),(118, 99, 120),(98, 99, 118),(100, 103, 121),(119, 103, 121),(117, 113, 119),(103, 100, 102),(121, 118, 99)
Created R_3: (115, 97, 100),(106, 104, 97),(115, 100, 104),(50, 49, 97),(115, 104, 100),(106, 107, 115),(97, 104, 120),(99, 110, 98),(115, 100, 117),(119, 119, 104)
Created R_4: (83, 68, 74),(75, 75, 74),(65, 83, 67),(78, 90, 88),(77, 115, 100),(119, 101, 105),(117, 115, 100),(83, 68, 83),(65, 85, 68),(67, 67, 67)
Created R_5: (100, 102, 115),(97, 100, 106),(101, 119, 114),(113, 119, 101),(102, 118, 99),(98, 110, 118),(98, 102, 102),(114, 116, 121),(104, 103, 110),(98, 110, 109)
Created R_6: (83, 68, 69),(87, 82, 71),(78, 66, 78),(71, 72, 71),(74, 72, 71),(71, 70, 68),(83, 65, 83),(68, 86, 66),(67, 86, 66),(78, 70, 71)
Created R_7: (83, 65, 68),(87, 81, 69),(83, 68, 70),(68, 71, 66),(78, 71, 70),(70, 68, 83),(65, 83, 70),(70, 68, 72),(71, 74, 71),(74, 74, 72)
Created R_8: (115, 100, 119),(113, 101, 101),(114, 102, 32),(100, 102, 100),(115, 103, 102),(100, 104, 103),(102, 104, 103),(102, 106, 103),(102, 110, 118),(98, 118, 118)
Created R_9: (115, 97, 100),(119, 113, 101),(114, 100, 102),(100, 103, 102),(103, 106, 110),(104, 103, 106),(103, 104, 106),(103, 103, 118),(102, 100, 100),(115, 100, 115)
Created R_10: (115, 97, 100),(99, 118, 118),(118, 98, 98),(100, 102, 115),(102, 97, 115),(100, 102, 97),(115, 102, 100),(101, 119, 119),(113, 102, 100),(115, 102, 100)
Created R_11: (49, 50, 52),(51, 53, 54),(52, 54, 53),(52, 51, 50),(49, 51, 52),(51, 53, 52),(51, 53, 52),(54, 54, 53),(55, 53, 54),(53, 52, 50)
Created R_12: (115, 100, 97),(100, 119, 113),(101, 119, 113),(114, 101, 114),(116, 116, 114),(104, 103, 118),(99, 120, 118),(99, 120, 118),(120, 99, 118),(99, 118, 118)
Created R_13: (97, 115, 100),(97, 97, 97),(97, 97, 97),(97, 97, 97),(97, 97, 115),(115, 115, 115),(115, 100, 100),(100, 100, 100),(100, 100, 100),(100, 100, 100)
Created R_14: (113, 119, 101),(119, 113, 101),(101, 119, 101),(114, 101, 114),(117, 102, 100),(121, 117, 102),(100, 115, 118),(99, 110, 118),(98, 100, 110),(109, 102, 100)
Created R_15: (97, 115, 100),(97, 115, 100),(103, 113, 119),(121, 101, 119),(104, 106, 100),(115, 110, 99),(120, 98, 110),(118, 102, 104),(100, 115, 102),(109, 110, 115)
^C
Someone pressed CTRL+C so the program terminates.
```

Output File



```
54.900000 6.600000 -3.500000 6.600000 61.500000 20.600000 -3.500000 20.600000 67.900000
90.700000 25.800000 16.200000 25.800000 61.000000 -16.700000 16.200000 -16.700000 87.300000
1129.100000 544.700000 329.900000 544.700000 740.000000 325.600000 329.900000 325.600000 790.500000
845.300000 262.700000 188.900000 262.700000 787.900000 795.000000 188.900000 795.000000 929.500000
371.300000 259.900000 160.700000 259.900000 310.100000 154.200000 160.700000 154.200000 221.800000
340.900000 349.900000 186.000000 349.900000 392.800000 197.600000 186.000000 197.600000 192.900000
59.700000 -26.100000 2.500000 -26.100000 37.800000 -23.200000 2.500000 -23.200000 58.100000
742.800000 554.400000 815.300000 554.400000 869.900000 711.500000 815.300000 711.500000 1077.900000
36.300000 28.000000 5.600000 28.000000 43.400000 -6.900000 5.600000 -6.900000 57.900000
31.400000 25.700000 18.900000 25.700000 52.100000 21.500000 18.900000 21.500000 43.800000
584.700000 595.400000 537.000000 595.400000 654.900000 562.300000 537.000000 562.300000 798.400000
787.100000 727.200000 562.900000 727.200000 672.800000 511.600000 562.900000 511.600000 757.100000
89.800000 37.400000 16.500000 37.400000 77.600000 27.100000 16.500000 27.100000 65.100000
77.400000 6.000000 38.500000 6.000000 37.200000 27.100000 38.500000 27.100000 64.000000
43.200000 1.600000 -26.700000 1.600000 65.300000 17.100000 -26.700000 17.100000 90.000000
298.500000 244.600000 -12.700000 244.600000 320.800000 27.000000 -12.700000 27.000000 345.800000
79.600000 5.100000 7.200000 5.100000 53.900000 3.200000 7.200000 3.200000 64.600000
578.200000 124.500000 55.600000 124.500000 365.800000 214.500000 55.600000 214.500000 240.900000
56.200000 -5.900000 8.800000 -5.900000 74.500000 2.500000 8.800000 2.500000 46.400000
69.100000 27.800000 -3.600000 27.800000 82.600000 9.100000 -3.600000 9.100000 76.600000
44.000000 31.900000 -22.000000 31.900000 41.800000 -27.600000 -22.000000 -27.600000 64.400000
39.600000 3.700000 0.500000 3.700000 63.200000 11.200000 0.500000 11.200000 57.400000
58.100000 -39.200000 -0.100000 -39.200000 56.400000 -9.800000 -0.100000 -9.800000 45.100000
67.800000 -6.000000 -8.300000 -6.000000 74.900000 -6.500000 -8.300000 -6.500000 77.900000
68.500000 -22.500000 -17.500000 -22.500000 78.500000 -4.900000 -17.500000 -4.900000 66.800000
65.800000 -7.700000 10.000000 -7.700000 50.100000 -30.600000 10.000000 -30.600000 73.600000
75.100000 37.800000 -9.900000 37.800000 570.500000 -75.700000 -9.900000 -75.700000 53.500000
86.600000 -9.900000 -29.000000 -9.900000 46.900000 -5.100000 -29.000000 -5.100000 55.200000
57.600000 -37.700000 -5.400000 -37.700000 52.600000 -6.500000 -5.400000 -6.500000 69.500000
61.900000 -24.700000 5.600000 -24.700000 46.100000 -13.800000 5.600000 -13.800000 22.500000
38.600000 -17.100000 -6.200000 -17.100000 27.700000 -7.100000 -6.200000 -7.100000 50.700000
47.100000 5.700000 -8.900000 5.700000 80.300000 13.200000 -8.900000 13.200000 68.800000
55.900000 -37.900000 -35.200000 -37.900000 81.700000 53.700000 -35.200000 53.700000 58.800000
73.600000 -24.900000 -25.300000 -24.900000 56.100000 13.700000 -25.300000 13.700000 62.000000
612.000000 -48.400000 65.600000 -48.400000 46.600000 23.500000 65.600000 23.500000 76.100000
45.300000 -5.900000 -17.800000 -5.900000 68.200000 20.100000 -17.800000 20.100000 74.800000
72.200000 -35.900000 1.000000 -35.900000 73.400000 -34.300000 1.000000 -34.300000 38.400000
401.800000 279.100000 117.000000 279.100000 642.100000 380.100000 117.000000 380.100000 406.300000
250.200000 159.300000 238.100000 159.300000 169.100000 182.800000 238.100000 182.800000 338.300000
886.600000 668.900000 344.300000 668.900000 1063.000000 622.900000 344.300000 622.900000 597.200000
53.200000 -21.900000 -29.300000 -21.900000 68.700000 14.700000 -29.300000 14.700000 85.700000
380.200000 270.000000 242.100000 270.000000 254.100000 184.200000 242.100000 184.200000 169.500000
59.600000 -4.000000 -20.200000 -4.000000 36.900000 20.500000 -20.200000 20.500000 56.500000
124.000000 104.500000 111.100000 104.500000 205.600000 170.000000 111.100000 170.000000 247.700000
43.000000 -3.900000 6.700000 -3.900000 9.500000 -8.900000 6.700000 -8.900000 49.500000
37.600000 -11.600000 -18.200000 -11.600000 54.600000 12.400000 -18.200000 12.400000 63.700000
86.200000 2.900000 -29.200000 2.900000 63.900000 13.500000 -29.200000 13.500000 38.600000
39.600000 -34.100000 -14.600000 -34.100000 67.200000 -5.900000 -14.600000 -5.900000 36.800000
61.700000 6.900000 -29.900000 6.900000 66.500000 -5.100000 -29.900000 -5.100000 77.900000
53.600000 -8.800000 -35.900000 -8.800000 51.600000 -4.500000 -35.900000 -4.500000 78.500000
315.700000 220.400000 109.300000 220.400000 258.100000 100.600000 109.300000 100.600000 260.500000
68.600000 -2.700000 22.800000 -2.700000 72.100000 -14.500000 22.800000 -14.500000 54.200000
```